Breve explicação do funcionamento de cada estrutura de dados

A estrutura de dados de lista simples é implementada no código fornecido como LinhaList. Esta estrutura consiste em nós de NodeLinhaList, onde cada nó representa uma Companhia e aponta para o próximo nó na lista. A primeira parte do código define a classe NodeLinhaList, que possui um atributo Companhia e um ponteiro next para o próximo nó.A LinhaList mantém um ponteiro para o primeiro nó firstNode na lista, o tamanho da lista size e um ponteiro para uma lista de paradas de ônibus associadas a essa linha, parada. Logo, a lista encadeada simples é usada para organizar as diferentes linhas de ônibus, e a lista de paradas está pontualmente associada a cada linha de ônibus.

Já a estrutura de dados de lista circular é implementada no código como hourRoundList. Nesta estrutura, cada nó (hourRoundNode) representa um horário de chegada e saída, bem como um nome associado, ou seja, cada nó contém informações sobre uma parada de ônibus. Como dito anteriormente a lista é circular, o que significa que o último nó aponta de volta para o primeiro, formando um ciclo e mantendo a sua circularidade. Cada nó tem um ponteiro next para o próximo nó e um ponteiro prev para o nó anterior .A hourRoundList mantém um ponteiro para o primeiro nó firstNode um ponteiro para o último nó lastNode na lista e também mantém o tamanho da lista size.

Para que Servem, em que Contexto Serão Utilizadas no Projeto?

As estruturas de dados definidas no códigotêm finalidades específicas, tais como A lista simples LinhaList serve para armazenar informações sobre diferentes companhias. Essa estrutura será proveitosa no projeto, pois manterá um registro de várias companhias que operam no sistema de transporte. Enquanto isso, a Estrutura hourRoundList é adequada para criar um ciclo de horários de chegada e saída, com os respectivos nomes associados. É significamente útil no agendamento da programação de horários que se repete ao longo do uso do menu. As estruturas de dados foram utilizadas para organizar e armazenar essas informações, permitindo operações como adicionar, remover e acessar paradas em linhas de ônibus.

Explicação das Três Operações Principais: Inserir, Remover e Acessar Valor

Operações-base do projeto:

➢ A função inserirNaListaSimples é capaz de inserir uma nova companhia no final da lista. Ela cria um novo nó NodeLinhaList com o nome da companhia e o adiciona à lista.
➢ A função apagarNoIndexListaSimples atua na remoção de um nó de índice específico, ajustando os ponteiros next para manter a integridade da lista e o seu uso em outras funções contribui para a otimização do tempo no decorrer do código.
➢ A função removeParada é usada para remover uma parada de uma linha de ônibus específica. Ela começa encontrando a linha desejada usando pegarDoIndexListaSimples e em seguida, verifica se essa linha tem uma lista de

paradas associadas. Se a linha não tiver paradas, uma mensagem informando que a linha não possui paradas é exibida.Se não, o algoritmo percorre a lista de paradas e remove a parada com o nome especificado pelo usuário. Se a parada for a primeira da lista, os ponteiros da lista de paradas são ajustados para remover a referência à primeira parada. Se a parada estiver em outra posição, a função percorre a lista de paradas e remove a parada desejada, ajustando os ponteiros apropriados.

➢ A função pegarDoIndexListaSimples permite acessar um nó da lista com base em seu índice. Isso facilita a recuperação de informações sobre uma companhia específica na lista.

➢ A função atribuirParadaEmLinha atribui uma nova parada a uma linha de ônibus específica. Ela começa criando um novo nó de parada e preenchendo suas informações. Em seguida, encontra a linha desejada usando pegarDoIndexListaSimples. Se a linha não tiver uma lista de paradas associada (parada), uma nova lista de paradas é criada. O novo nó de parada é então inserido na lista de paradas da linha usando a função inserirNaListaCircular. Isso permite que novas paradas sejam atribuídas a uma linha de ônibus existente.

➢ A função inserirNaListaCircular terá a função de inserir um novo horário com nome associado no final da lista circular. Ela atualiza os ponteiros next e prev para manter a circularidade da lista.
Tanto as estruturas de dados criadas como as operações definida são importantes para o armazenamento e manipulação eficiente de dados e informações no projeto, os quais foramusados para a listagem de companhias e programação de horários

➢ A função ImprimirLinhascomParada será essencial já que percorrerá a lista de linhas de ônibus, verificando se cada linha tem uma lista de paradas. Para cada linha com uma lista de paradas, ele imprime o nome da linha e, em seguida, chama a função ImprimirListaCircular para imprimir a lista de paradas associada a essa linha. Se a linha não tiver uma lista de paradas, uma mensagem informando que a linha não possui paradas é exibida para o maior esclarecimento possível do usuário.

A função alterarParada está correlacionada a uma função básica de acesso e essa função recebe como entrada uma lista de linhas de ônibus, um índice de linha, o nome atual de uma parada e o novo nome que se deseja atribuir à parada. Ela realiza a seguinte sequência de ações: primeiro, verifica se o índice da linha é válido e se a linha existe; em seguida, verifica se a linha possui uma lista de paradas associada; se essas condições são atendidas, a função itera pelas paradas da linha em busca da parada com o nome especificado. Quando a parada é encontrada, o nome é alterado para o novo nome fornecido. Em caso de sucesso, a função exibe uma mensagem informando que a operação foi bem-sucedida. Se a parada com o nome especificado não for encontrada, a função exibe uma mensagem de erro indicando que a parada não foi localizada na lista de paradas da linha.